

# Numerical Abstract Domain using Support Function.

Yassamine Seladji and Olivier Bouissou.



CEA, LIST, LMeASI. France  
[yassamine.seladji@cea.fr](mailto:yassamine.seladji@cea.fr)  
[olivier.bouissou@cea.fr](mailto:olivier.bouissou@cea.fr)

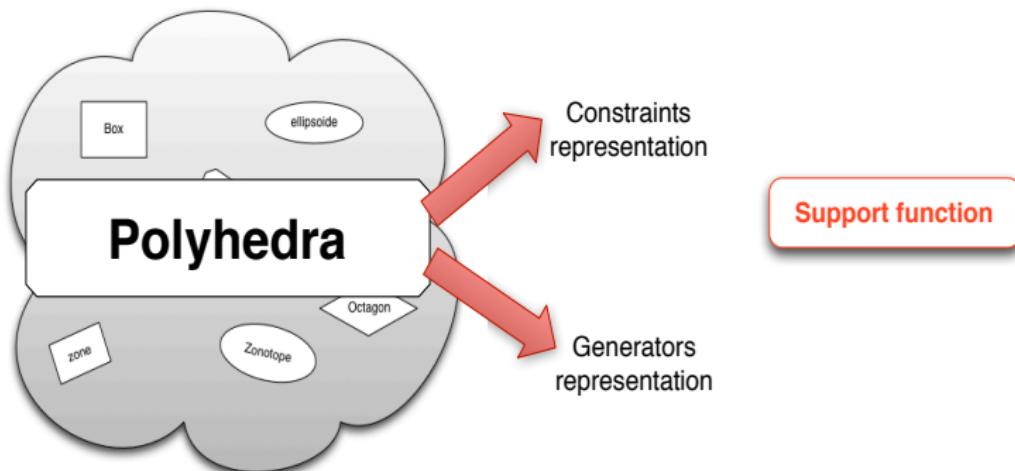
## Abstract Numerical Domain

## Abstract Numerical Domain

## Abstract Numerical Domain

$$\mathbb{P} = \bigwedge_{i \in [1, m]} A_i x \leq b_i$$

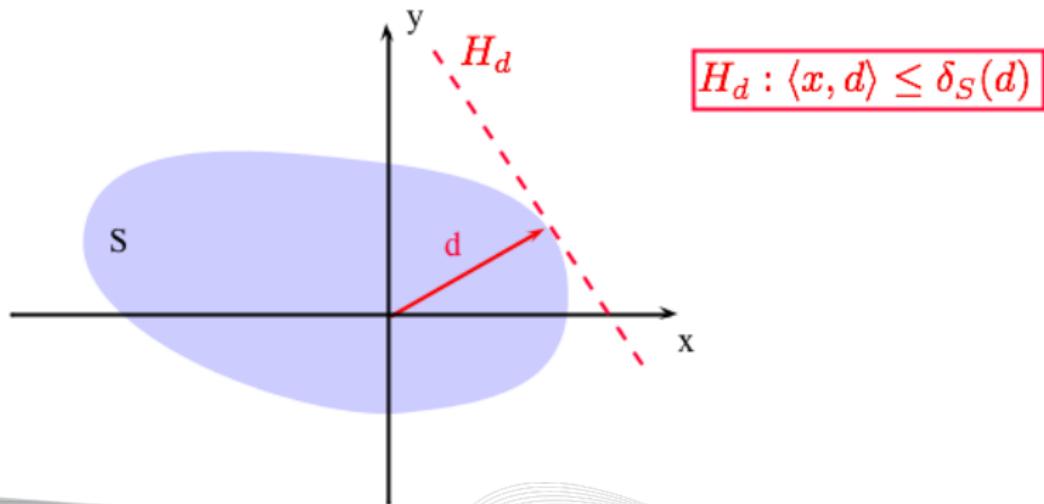
$$\mathbb{P} = co\{p_1, \dots, p_n\} + cone\{r_1, \dots, r_m\}$$



### Definition

Let  $S$  be a closed convex set. The support function of  $S$ , noted  $\delta_S$ , is defined as :

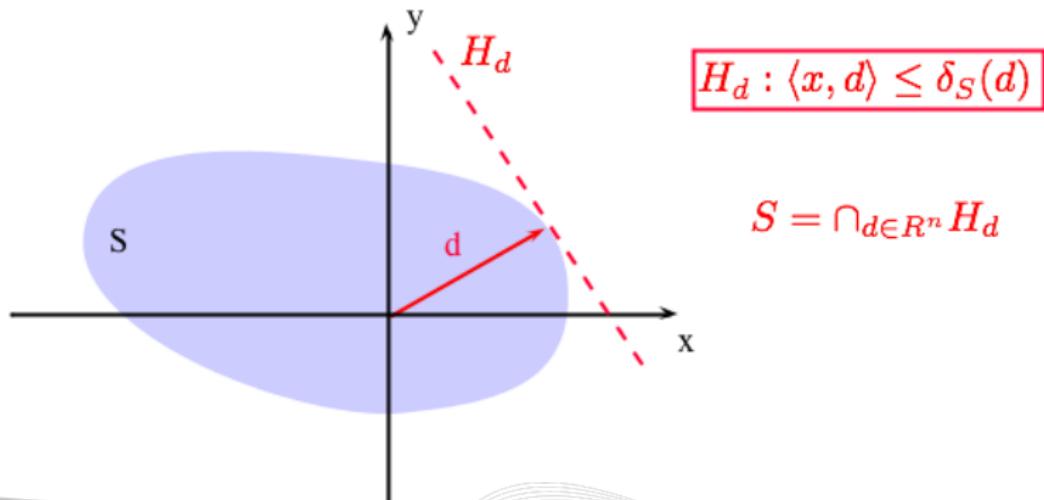
$$\forall d \in \mathbb{R}^n, \delta_S(d) = \sup\{\langle x, d \rangle : x \in S\}$$



### Definition

Let  $S$  be a closed convex set. The support function of  $S$ , noted  $\delta_S$ , is defined as :

$$\forall d \in \mathbb{R}^n, \delta_S(d) = \sup\{\langle x, d \rangle : x \in S\}$$



### The special case of polyhedron

Let  $S$  be a polyhedron. If  $S$  is represented by :

- ▶ Constraints system,  $\delta_S$  is obtained using Linear Programming.
- ▶ Generators,

### The special case of polyhedron

Let  $S$  be a polyhedron. If  $S$  is represented by :

- ▶ Constraints system,  $\delta_S$  is obtained using Linear Programming.

$$\max \langle x, d \rangle$$

$$\text{st} : \bigwedge_{i \in [1, m]} A_i x \leq b_i$$

- ▶ Generators,

## The special case of polyhedron

Let  $S$  be a polyhedron. If  $S$  is represented by :

- ▶ Constraints system,  $\delta_S$  is obtained using Linear Programming.
- ▶ Generators,

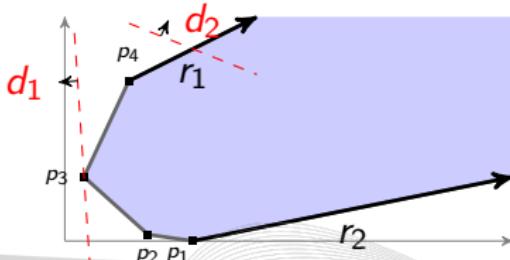
$$\forall d \in n, \delta_{\mathbb{P}}(d) = \begin{cases} \max_{i \in [1, k]} \langle p_i, d \rangle & \text{if } \forall j \in [1, l], \langle r_j, d \rangle \leq 0 \\ +\infty & \text{otherwise} \end{cases}.$$

## The special case of polyhedron

Let  $S$  be a polyhedron. If  $S$  is represented by :

- ▶ Constraints system,  $\delta_S$  is obtained using Linear Programming.
- ▶ Generators,

$$\forall d \in n, \delta_{\mathbb{P}}(d) = \begin{cases} \max_{i \in [1, k]} \langle p_i, d \rangle & \text{if } \forall j \in [1, l], \langle r_j, d \rangle \leq 0 \\ +\infty & \text{otherwise} \end{cases}.$$



Let  $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$  be a set of directions.

Let  $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$  be a set of directions.



Let  $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$  be a set of directions.

### Property

We have that :

$$P = \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n | \langle x, d \rangle \leq \delta_S(d)\}$$

Then

$$S \subseteq P$$



Let  $\Delta = \{d_1, d_2, d_3, d_4, d_5\}$  be a set of directions.

### Property

We have that :

$$P = \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n | \langle x, d \rangle \leq \delta_S(d)\}$$

Then

$$S \subseteq P$$

## Properties

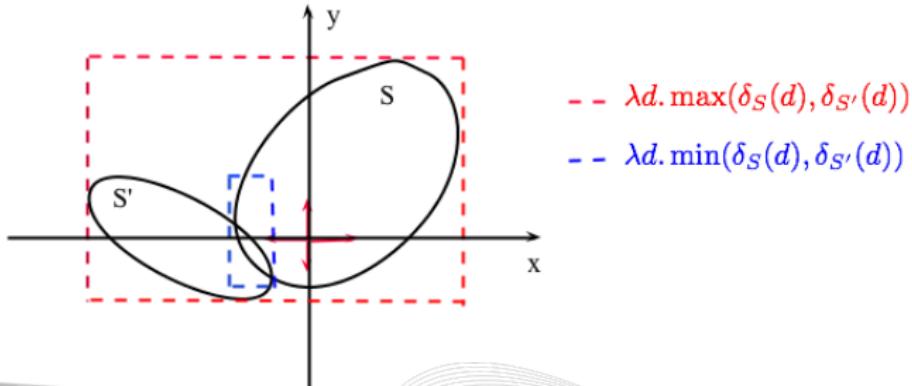
Let  $S, S'$  be two closed convex sets. We have :

- ▶  $\forall M \in \mathbb{R}^{n \times m}, \delta_{MS}(d) = \delta_S(M^T d).$
- ▶  $\delta_{S \oplus S'}(d) = \delta_S(d) + \delta_{S'}(d).$   $S \oplus S' = \{x + x' \mid x \in S, x' \in S'\}$
- ▶  $\delta_{S \cup S'}(d) = \max(\delta_S(d), \delta_{S'}(d)).$
- ▶  $\delta_{S \cap S'}(d) \leq \min(\delta_S(d), \delta_{S'}(d)).$

## Properties

Let  $S, S'$  be two closed convex sets. We have :

- ▶  $\forall M \in \mathbb{R}^{n \times m}, \delta_{MS}(d) = \delta_S(M^T d).$
- ▶  $\delta_{S \oplus S'}(d) = \delta_S(d) + \delta_{S'}(d).$   $S \oplus S' = \{x + x' \mid x \in S, x' \in S'\}$
- ▶  $\delta_{S \cup S'}(d) = \max(\delta_S(d), \delta_{S'}(d)).$
- ▶  $\delta_{S \cap S'}(d) \leq \min(\delta_S(d), \delta_{S'}(d)).$



For a set of directions  $\Delta$ ,

For a set of directions  $\Delta$ , let  $\mathbb{P}_\Delta^\sharp = \Delta \rightarrow \mathbb{R}_\infty$  be the abstract domain.

For a set of directions  $\Delta$ , let  $\mathbb{P}_\Delta^\sharp = \Delta \rightarrow \mathbb{R}_\infty$  be the abstract domain.

### The concretisation function

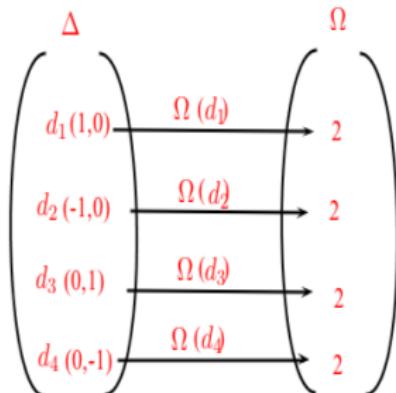
$$\begin{array}{rccc} \gamma_\Delta : & (\Delta \rightarrow \mathbb{R}_\infty) & \longrightarrow & P(\mathbb{R}^n) \\ & \Omega & \longrightarrow & \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n \mid \langle x, d \rangle \leq \Omega(d)\} \end{array}$$

For a set of directions  $\Delta$ , let  $\mathbb{P}_\Delta^\sharp = \Delta \rightarrow \mathbb{R}_\infty$  be the abstract domain.

### The concretisation function

$$\begin{aligned}\gamma_\Delta : (\Delta \rightarrow \mathbb{R}_\infty) &\longrightarrow P(\mathbb{R}^n) \\ \Omega &\longrightarrow \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n \mid \langle x, d \rangle \leq \Omega(d)\}\end{aligned}$$

Example :

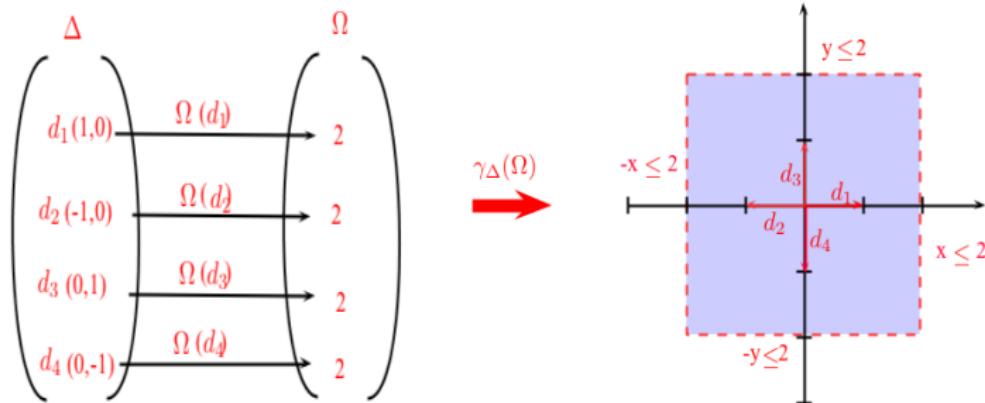


For a set of directions  $\Delta$ , let  $\mathbb{P}_\Delta^\sharp = \Delta \rightarrow \mathbb{R}_\infty$  be the abstract domain.

### The concretisation function

$$\begin{aligned}\gamma_\Delta : (\Delta \rightarrow \mathbb{R}_\infty) &\longrightarrow P(\mathbb{R}^n) \\ \Omega &\longrightarrow \bigcap_{d \in \Delta} \{x \in \mathbb{R}^n \mid \langle x, d \rangle \leq \Omega(d)\}\end{aligned}$$

Example :



## The abstraction function

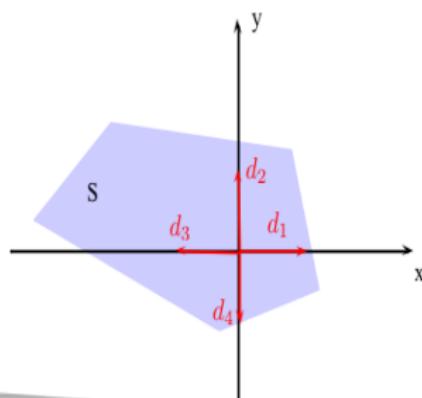
$$\begin{aligned}\alpha_\Delta : P(\mathbb{R}^n) &\longrightarrow (\Delta \rightarrow \mathbb{R}_\infty) \\ S &\longrightarrow \begin{cases} \lambda d. -\infty & \text{if } S = \emptyset \\ \lambda d. +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d. \delta_S(d) & \text{otherwise} \end{cases}\end{aligned}$$

Example :

## The abstraction function

$$\alpha_{\Delta} : P(\mathbb{R}^n) \rightarrow (\Delta \rightarrow \mathbb{R}_{\infty})$$
$$S \rightarrow \begin{cases} \lambda d. -\infty & \text{if } S = \emptyset \\ \lambda d. +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d. \delta_S(d) & \text{otherwise} \end{cases}$$

Example :

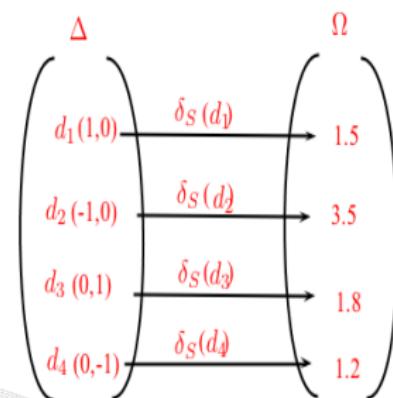
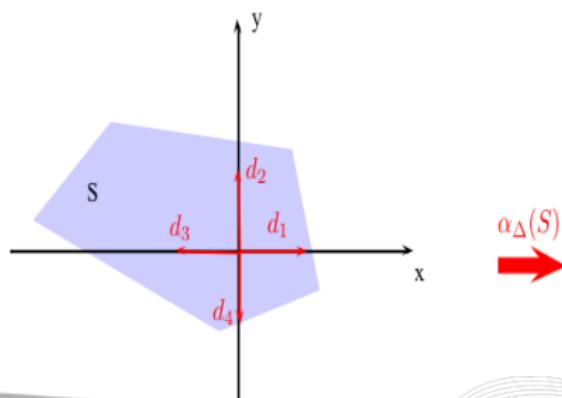


## The abstraction function

$$\alpha_{\Delta} : P(\mathbb{R}^n) \rightarrow (\Delta \rightarrow \mathbb{R}_{\infty})$$

$$S \rightarrow \begin{cases} \lambda d. -\infty & \text{if } S = \emptyset \\ \lambda d. +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d. \delta_S(d) & \text{otherwise} \end{cases}$$

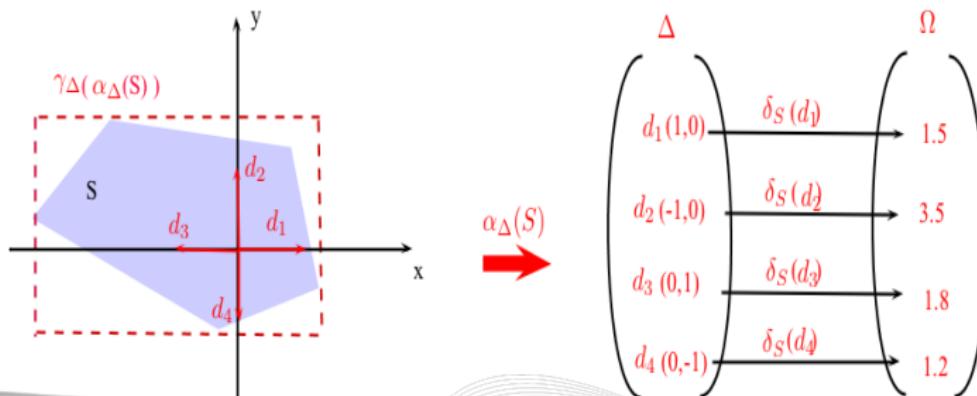
Example :



## The abstraction function

$$\alpha_{\Delta} : \begin{aligned} P(\mathbb{R}^n) &\longrightarrow (\Delta \rightarrow \mathbb{R}_{\infty}) \\ S &\longrightarrow \begin{cases} \lambda d. -\infty & \text{if } S = \emptyset \\ \lambda d. +\infty & \text{if } S = \mathbb{R}^n \\ \lambda d. \delta_S(d) & \text{otherwise} \end{cases} \end{aligned}$$

Example :



The complete lattice  $\langle \mathbb{P}_\Delta^\sharp, \sqsubseteq, \perp, \top, \sqcup, \sqcap \rangle$  is defined by :

- ▶ An order relation :  $\Omega_1 \sqsubseteq \Omega_2 \Leftrightarrow \gamma_\Delta(\Omega_1) \subseteq \gamma_\Delta(\Omega_2)$ .
- ▶ A minimal element :  $\perp = \lambda d. -\infty$ .
- ▶ A maximal element :  $\top = \lambda d. +\infty$ .
- ▶ A join operator :  $\Omega_1 \sqcup \Omega_2 = \lambda d. \max(\Omega_1(d), \Omega_2(d))$ .
- ▶ A meet operator :  $\Omega_1 \sqcap \Omega_2 = \lambda d. \min(\Omega_1(d), \Omega_2(d))$ .



The complete lattice  $\langle \mathbb{P}_\Delta^\sharp, \sqsubseteq, \perp, \top, \sqcup, \sqcap \rangle$  is defined by :

- ▶ An order relation :  $\Omega_1 \sqsubseteq \Omega_2 \Leftrightarrow \gamma_\Delta(\Omega_1) \subseteq \gamma_\Delta(\Omega_2)$ .
- ▶ A minimal element :  $\perp = \lambda d. -\infty$ .
- ▶ A maximal element :  $\top = \lambda d. +\infty$ .
- ▶ A join operator :  $\Omega_1 \sqcup \Omega_2 = \lambda d. \max(\Omega_1(d), \Omega_2(d))$ .
- ▶ A meet operator :  $\Omega_1 \sqcap \Omega_2 = \lambda d. \min(\Omega_1(d), \Omega_2(d))$ .

Property :

$$\begin{aligned}\gamma_\Delta(\Omega_1 \sqcup \Omega_2) &= \gamma_\Delta(\Omega_1) \cup \gamma_\Delta(\Omega_2). \\ \gamma_\Delta(\Omega_1 \sqcap \Omega_2) &\supseteq \gamma_\Delta(\Omega_1) \cap \gamma_\Delta(\Omega_2).\end{aligned}$$



**Template abstract domain** : S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In VMCAI. Springer, 2005.

### Similarities :

- ▶ Abstract domain based on a static choice of directions set.
- ▶ The same definition of inclusion, meet and join operators.



**Template abstract domain** : S. Sankaranarayanan, H. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In VMCAI. Springer, 2005.

### Similarities :

- ▶ Abstract domain based on a static choice of directions set.
- ▶ The same definition of inclusion, meet and join operators.

### Differences :

The fixpoint computation

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^m$ .

$X \in \mathbb{P}_0$

$X = AX + b$ .

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^m$ .

$X \in \mathbb{P}_0$

$X = AX + b$ .

## The abstract element

$$\Omega = \lambda d. \delta_{A\mathbb{P}_0 \oplus b}(d)$$



## Affine Computation : Affine transformation

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^m$ .

$X \in \mathbb{P}_0$

$X = AX + b$ .

## The abstract element

$$\begin{aligned}\Omega &= \lambda d. \delta_{A\mathbb{P}_0 \oplus b}(d) \\ &= \lambda d. \delta_{\mathbb{P}_0}(A^T d) + \langle b, d \rangle\end{aligned}$$

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^m$ .

$X \in \mathbb{P}_0$

$X = AX + b$ .

## The abstract element

$$\begin{aligned}\Omega &= \lambda d. \delta_{A\mathbb{P}_0 \oplus b}(d) \\ &= \lambda d. \delta_{\mathbb{P}_0}(A^T d) + \langle b, d \rangle\end{aligned}$$

$$\Omega = \alpha_\Delta(A\mathbb{P}_0 \oplus b).$$



## Affine computation : Kleene fixpoint computation

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^n$ .

Input :  $c \in \mathbb{R}^n$ ,  $l \in \mathbb{R}$

$X \in \mathbb{P}_0$

while ( $\langle X, c \rangle \leq l$ ) {

$X = AX + b$ .

}



## Affine computation : Kleene fixpoint computation

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^n$ .

Input :  $c \in \mathbb{R}^n$ ,  $l \in \mathbb{R}$

$X \in \mathbb{P}_0$

while ( $\langle X, c \rangle \leq l$ ) {

$X = AX + b$ .

}

$$\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \cap (\langle c, X \rangle \leq l)]$$



## Affine computation : Kleene fixpoint computation

- ▶ **Case 1 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \cap (\{c_i/X\} / \{/\})]$

## Program

```
X ∈ P₀
while (true) {
    X = AX + b
}
```



## Affine computation : Kleene fixpoint computation

- ▶ **Case 1 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \cap (\{c_i/X\} / \{/\})]$

## Program

```
X ∈ ℙ₀  
while (true) {  
    X = AX + b  
}
```

## The first abstract element

$$\Omega_1 = \lambda d. \delta_{\mathbb{P}_0 \cup (A\mathbb{P}_0 \oplus b)}(d)$$



## Affine computation : Kleene fixpoint computation

- ▶ **Case 1 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \cap (\{c_i/X\} / \{/\})]$

## Program

```
X ∈ P₀  
while (true) {  
    X = AX + b  
}
```

## The first abstract element

$$\begin{aligned}\Omega_1 &= \lambda d. \delta_{P_0 \cup (AP_0 \oplus b)}(d) \\ &= \lambda d. \max(\delta_{P_0}(d), \delta_{P_0}(A^T d) + \langle b, d \rangle)\end{aligned}$$



## Affine computation : Kleene fixpoint computation

- ▶ **Case 1 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \cap (\{c_i/X\} / \#/\#)]$

## Program

```
X ∈ P₀
while (true) {
    X = AX + b
}
```

## The first abstract element

$$\begin{aligned}\Omega_1 &= \lambda d. \delta_{P_0 \cup (AP_0 \oplus b)}(d) \\ &= \lambda d. \max(\delta_{P_0}(d), \delta_{P_0}(A^T d) + \langle b, d \rangle)\end{aligned}$$

The  $i^{th}$  abstract element

$$\Omega_i = \lambda d. \max\left(\delta_{P_0}(d), \max_{j \in [1, i]} \left( \delta_{P_0}(A^{Tj} d) + \sum_{k=1}^j \delta_{P_b}(A^{T(k-1)} d) \right)\right)$$



## Affine computation : Kleene fixpoint computation

- ▶ **Case 1 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \cap (\{c_i/X\} / \{/\})]$

## Program

```
X ∈ ℙ₀
while (true) {
    X = AX + b
}
```

$$\alpha_\Delta(\mathbb{P}_i) = \Omega_i$$

The  $i^{th}$  abstract element

$$\Omega_i = \lambda d. \max \left( \delta_{\mathbb{P}_0}(d), \max_{j \in [1, i]} \left( \delta_{\mathbb{P}_0}(A^{Tj} d) + \sum_{k=1}^j \delta_{\mathbb{P}_b}(A^{T(k-1)} d) \right) \right)$$



## Affine computation : Kleene fixpoint computation

$$\Omega_i = \lambda d. \max\left(\delta_{\mathbb{P}_0}(d), \max_{j \in [1, i]} \left( \delta_{\mathbb{P}_0}(A^T j d) + \sum_{k=1}^j \delta_{\mathbb{P}_b}(A^{T(k-1)} d) \right)\right)$$

**Algorithm 1** Kleene Algorithm using support function.

**Require:**  $\Delta \subset \mathbb{R}^n$ , set of  $l$  directions.  $\mathbb{P}_0$ , The initial polyhedron

**Require:**  $A \in \mathbb{R}^n \times \mathbb{R}^m$ ,  $b \in \mathbb{R}^m$

```

1:  $D = \Delta$ ,  $\Omega = \delta_{\mathbb{P}_0}(\Delta)$ 
2: repeat
3:    $\Omega' = \Omega$ 
4:   for all  $i = 0, \dots, (l - 1)$  do
5:      $\Theta[i] = \Theta[i] + \langle b, D[i] \rangle$ 
6:      $D[i] = A^T D[i]$ 
7:      $\Upsilon[i] = \delta_{\mathbb{P}_0}(D[i]) + \Theta[i]$ 
8:      $\Omega[i] = \max(\Omega[i], \Upsilon[i])$ 
9:   end for
10:  until  $\Omega \sqsubseteq \Omega'$ 

```



## Affine computation : Kleene fixpoint computation

- ▶ **Case 2 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq I)]$

**Program**

```
X ∈ ℙ₀
while (⟨X, c⟩ ≤ I)
{
    X = AX + b
}
```



## Affine computation : Kleene fixpoint computation

- ▶ **Case 2 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq I)]$

**Program**

```
X ∈ ℙ₀
while (⟨X, c⟩ ≤ I)
{
    X = AX + b
}
```

$$H : \langle X, c \rangle \leq I$$



## Affine computation : Kleene fixpoint computation

- ▶ **Case 2 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq I)]$

**Program**

```
X ∈ ℙ₀
while (⟨X, c⟩ ≤ I)
{
    X = AX + b
}
```

$$H : \langle X, c \rangle \leq I$$

$$\delta_H(d) = \begin{cases} I & \text{if } d = \lambda c, \lambda \geq 0 \\ +\infty & \text{otherwise} \end{cases}$$



## Affine computation : Kleene fixpoint computation

- ▶ **Case 2 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq I)]$

**Program**

```
X ∈ ℙ₀
while (⟨X, c⟩ ≤ I)
{
    X = AX + b
}
```

$$H : \langle X, c \rangle \leq I$$

$$\delta_H(d) = \begin{cases} I & \text{if } d = \lambda c, \lambda \geq 0 \\ +\infty & \text{otherwise} \end{cases}$$

We put  $\Delta \cup \{c\}$  :

- ▶  $\Delta_1 = \{c\}$  .
- ▶  $\Delta_2 = \Delta \setminus \{d \in \Delta \mid d = \lambda c, \lambda \geq 0\}$

- ▶ **Case 2 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$
- ▶ For  $d \in \Delta_2$  : we use the same method as for **Case 1**.

- ▶ **Case 2 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$
- ▶ For  $d \in \Delta_2$  : we use the same method as for **Case 1**.

$$\Omega_i = \lambda d. \max \left( \delta_{\mathbb{P}_0}(d), \max_{j \in [1, i]} \left( \delta_{\mathbb{P}_0}(A^{Tj}d) + \sum_{k=1}^j \delta_{\mathbb{P}_b}(A^{T(k-1)}d) \right) \right)$$



## Affine computation : Kleene fixpoint computation

- ▶ **Case 2 :**  $\Omega_i = \Omega_{i-1} \sqcup [(A\Omega_{i-1} + b) \sqcap (\langle c, X \rangle \leq l)]$
- ▶ For  $d \in \Delta_2$  : we use the same method as for **Case 1**.

$$\Omega_i = \lambda d. \max \left( \delta_{\mathbb{P}_0}(d), \max_{j \in [1, i]} \left( \delta_{\mathbb{P}_0}(A^{Tj}d) + \sum_{k=1}^j \delta_{\mathbb{P}_b}(A^{T(k-1)}d) \right) \right)$$

- ▶ For  $d \in \Delta_1$ , we have that :

$$\Omega_i = \lambda d. \max(\delta_{\gamma_\Delta(\Omega_{i-1})}(d), \min(\delta_{\gamma_\Delta(\Omega_{i-1})}(A^T d) + \langle b, d \rangle, l))$$

Such that  $\lambda d. \delta_{\mathbb{P}_i}(d) \leq \Omega_i(d)$ .

## Non-linear computation : non-linear transformation

### Non-linear Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $f$  non-linear function.

$X \in \mathbb{P}_0$

$X = f(X)$

## Non-linear computation : non-linear transformation

## Non-linear Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $f$  non-linear function.

$X \in \mathbb{P}_0$

$X = f(X)$



## Linear Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$



## Non-linear computation : non-linear transformation

## Non-linear Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $f$  non-linear function.

$X \in \mathbb{P}_0$

$X = f(X)$



## Linear Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

## The Linearisation Function

Let  $L$  be the Linearisation function, such that :

$$\begin{aligned} L : (\mathbb{R}^n \Rightarrow \mathbb{R}^n) \times \mathbb{P}_\Delta^\# &\Rightarrow (\mathbb{R}^n \Rightarrow \mathbb{I}_{\mathbb{R}}^n) \\ (f, \Omega) &\Rightarrow \mathbf{f} : \forall x \in \gamma_\Delta(\Omega), f(x) \in \mathbf{f}(x) \end{aligned}$$



## Non-linear computation : non-linear transformation

## The Linearisation Function

Let  $L$  be the Linearisation function, such that :

$$\begin{aligned} L : (\mathbb{R}^n \Rightarrow \mathbb{R}^n) \times \mathbb{P}_\Delta^\sharp &\Rightarrow (\mathbb{R}^n \Rightarrow \mathbb{I}_{\mathbb{R}}^n) \\ (f, \Omega) &\Rightarrow \mathbf{f} : \forall x \in \gamma_\Delta(\Omega), f(x) \in \mathbf{f}(x) \end{aligned}$$

Example :

Let  $f(x) = a * x^n$ , with  $a \in \mathbb{R}$  and  $\Omega \in \mathbb{P}_\Delta^\sharp$ .

$$L(a * x^n, \Omega) = \underbrace{a * [\Omega(-x), \Omega(+x)]^{n-1}}_a * x$$

We obtain the linear transformation  $\mathbf{a} * x$  with  $\mathbf{a} \in \mathbb{I}_{\mathbb{R}}$ .

## Program

Input :  $P_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in P_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$



## Non-linear computation : non-linear transformation

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

$$\Omega = \lambda d. \delta_{\mathbb{P}_0}(\mathbf{A}^T d) + \langle \mathbf{b}, d \rangle$$

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

$$\Omega = \lambda d. \delta_{\mathbb{P}_0}(\mathbf{A}^T d) + \langle \mathbf{b}, d \rangle$$

### Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

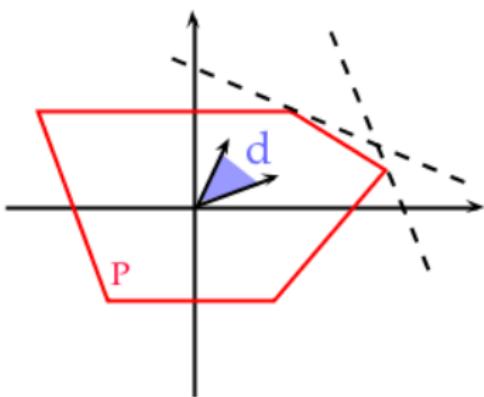
$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

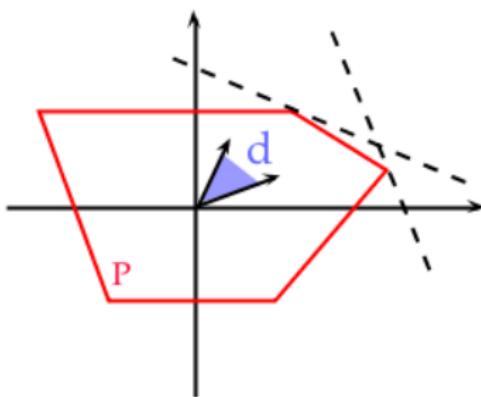
$$\Omega = \lambda d. \delta_{\mathbb{P}_0}(\mathbf{A}^T d) + \langle \mathbf{b}, d \rangle$$

### Interval Based Support Function

## Non-linear computation : non-linear transformation

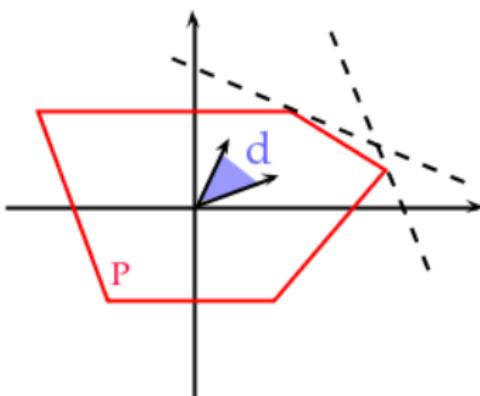


## Non-linear computation : non-linear transformation



$$\forall \mathbf{d} \in \mathbb{I}_n, \quad ? \leq \{\delta_{\mathbb{P}}(d) | d \in \mathbf{d}\} \leq ?$$

## Non-linear computation : non-linear transformation



$$\begin{aligned} \forall \mathbf{d} \in \mathbb{I}_n, \quad ? &\leq \{\delta_{\mathbb{P}}(d) | d \in \mathbf{d}\} & \leq ? \\ ? &\leq \{ \max_{i=1,\dots,5} \langle v_i, d \rangle | d \in \mathbf{d} \} & \leq ? \end{aligned}$$



## Non-linear computation : non-linear transformation

## Interval Based Support Function

Let  $\mathbb{P}$  be a polyhedron and  $\delta_{\mathbb{P}}$  be its support function. Let  $\mathbf{d} \in \mathbb{I}_n$  be an interval vector, representing a set of possible directions. We have that :

$$\forall d \in \mathbf{d}, \iota_{\mathbb{P}}(\mathbf{d}) \leq \delta_{\mathbb{P}}(d) \leq \sigma_{\mathbb{P}}(\mathbf{d})$$



## Non-linear computation : non-linear transformation

## Interval Based Support Function

Let  $\mathbb{P}$  be a polyhedron and  $\delta_{\mathbb{P}}$  be its support function. Let  $\mathbf{d} \in \mathbb{I}_n$  be an interval vector, representing a set of possible directions. We have that :

$$\forall d \in \mathbf{d}, \iota_{\mathbb{P}}(\mathbf{d}) \leq \delta_{\mathbb{P}}(d) \leq \sigma_{\mathbb{P}}(\mathbf{d})$$

## Interval Based Support Function Upper Bound :

$$\sigma_{\mathbb{P}} : \begin{cases} \mathbb{I}_n & \rightarrow \mathbb{R} \\ \mathbf{d} & \mapsto \begin{cases} \max_{i \in [1, k]} \overline{\langle v_i, \mathbf{d} \rangle} & \text{if } \forall j \in [1, l], \langle r_j, \mathbf{d} \rangle \cap ]0, +\infty[ = \emptyset \\ +\infty & \text{otherwise} \end{cases} \end{cases} .$$



## Non-linear computation : non-linear transformation

## Interval Based Support Function

Let  $\mathbb{P}$  be a polyhedron and  $\delta_{\mathbb{P}}$  be its support function. Let  $\mathbf{d} \in \mathbb{I}_n$  be an interval vector, representing a set of possible directions. We have that :

$$\forall d \in \mathbf{d}, \iota_{\mathbb{P}}(\mathbf{d}) \leq \delta_{\mathbb{P}}(d) \leq \sigma_{\mathbb{P}}(\mathbf{d})$$

## Interval Based Support Function Upper Bound :

$$\sigma_{\mathbb{P}} : \begin{cases} \mathbb{I}_n & \rightarrow \mathbb{R} \\ \mathbf{d} & \mapsto \begin{cases} \max_{i \in [1, k]} \overline{\langle v_i, \mathbf{d} \rangle} & \text{if } \forall j \in [1, l], \langle r_j, \mathbf{d} \rangle \cap ]0, +\infty[ = \emptyset \\ +\infty & \text{otherwise} \end{cases} \end{cases}$$

## Interval Based Support Function Lower Bound :

$$\iota_{\mathbb{P}} : \begin{cases} \mathbb{I}_n & \rightarrow \mathbb{R} \\ \mathbf{d} & \mapsto \max_{i \in [1, k]} \underline{\langle v_i, \mathbf{d} \rangle} \end{cases}$$

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

$$\Omega = \lambda d. \delta_{\mathbb{P}_0}(\mathbf{A}^T d) + \langle \mathbf{b}, d \rangle$$

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

$$\Omega = \lambda d. \delta_{\mathbb{P}_0} (\mathbf{A}^T d) + \langle \mathbf{b}, d \rangle$$

$$\Omega = \lambda d. \sigma_{\mathbb{P}_0} (\mathbf{A}^T d) + \overline{\langle \mathbf{b}, d \rangle}$$



## Non-linear computation : non-linear transformation

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

$$\Omega = \lambda d. \delta_{\mathbb{P}_0}(\mathbf{A}^T d) + \langle \mathbf{b}, d \rangle$$

$$\begin{aligned}\Omega &= \lambda d. \sigma_{\mathbb{P}_0}(\mathbf{A}^T d) + \overline{\langle \mathbf{b}, d \rangle} \\ &\geq \iota_{\mathbb{P}_0}(\mathbf{A}^T d) + \underline{\langle \mathbf{b}, d \rangle}\end{aligned}$$

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $\mathbf{A} \in \mathbb{I}_{\mathbb{R}}^{n \times n}$ ,  $\mathbf{b} \in \mathbb{I}_{\mathbb{R}}^n$ .

$X \in \mathbb{P}_0$

$X = \mathbf{A}X +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}$

$$\Omega = \lambda d. \delta_{\mathbb{P}_0} (\mathbf{A}^T d) + \langle \mathbf{b}, d \rangle$$

$$\left. \begin{array}{lcl} \Omega & = & \lambda d. \sigma_{\mathbb{P}_0} (\mathbf{A}^T d) + \overline{\langle \mathbf{b}, d \rangle} \\ & \geq & \iota_{\mathbb{P}_0} (\mathbf{A}^T d) + \underline{\langle \mathbf{b}, d \rangle} \end{array} \right\} \implies \textcolor{red}{\alpha_{\Delta}(\mathbf{A}\mathbb{P}_0 +_{\mathbb{I}_{\mathbb{R}}} \mathbf{b}) \sqsubseteq_{\Delta} \Omega}$$

## Program

Input :  $\mathbb{P}_0$  a polyhedron.

Input :  $f$  and  $g$  two non-linear function in  $\mathbb{R}^n$ .

$X \in \mathbb{P}_0$ ,  $\Omega_0 = \alpha_{\Delta}(\mathbb{P}_0)$ ,  $i = 0$

while ( $L(g, \Omega_i) \leq_{\mathbb{R}} \mathbf{l}$ ) {

$X = L(f, \Omega_i) +_{\mathbb{R}} \mathbf{b}$

$i++$

$$\Omega_i = \Omega_{i-1} \sqcup \left[ \left( L(f, \Omega_{i-1}) +_{\mathbb{R}} \mathbf{b} \right) \cap \left( L(g, \Omega_{i-1}) \leq_{\mathbb{R}} \mathbf{l} \right) \right]$$

## Fixpoint computation using Kleene iteration

The Algorithm doesn't guarantee the termination of the computation.



## Fixpoint computation using Kleene iteration

The Algorithm doesn't guarantee the termination of the computation.

- ▶ Solution 1 :

### widening

$$\forall \Omega_1, \Omega_2 \in \mathbb{P}_\Delta^\sharp, \Omega_1 \nabla_\Delta \Omega_2 = \lambda d. \begin{cases} \Omega_2(d) & \text{if } \Omega_1(d) = \Omega_2(d) \\ +\infty & \text{otherwise} \end{cases}$$



## Fixpoint computation using Kleene iteration

The Algorithm doesn't guarantee the termination of the computation.

▶ Solution 1 :

## widening

$$\forall \Omega_1, \Omega_2 \in \mathbb{P}_\Delta^\sharp, \Omega_1 \nabla_\Delta \Omega_2 = \lambda d. \begin{cases} \Omega_2(d) & \text{if } \Omega_1(d) = \Omega_2(d) \\ +\infty & \text{otherwise} \end{cases}$$

▶ Solution 2 :**The accelerated Kleene iteration [1]**

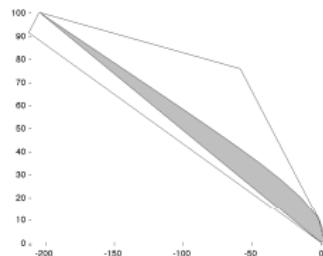
[1] Olivier Bouissou, Yassamine Seladji, and Alexandre Chapoutot. Acceleration of the abstract fixpoint computation in numerical program analysis. *Journal of Symbolic Computation*, 2011.



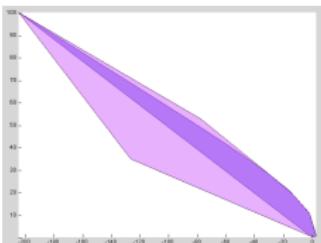
# Experimentation

## Kleene Algorithm using support function

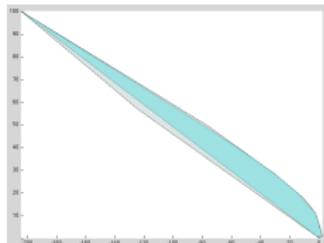
```
begin
    while (true) do
        xn = 0.5 *x - y - 2.5;
        yn = 0.9 *y + 10;
        x = xn; y = yn;
    done;
end
```



8 directions  
(0.044 seconds)



50 directions  
(0.34 seconds)



100 directions  
(0.7 seconds)

## Accelerated Kleene Algorithm using support function

```
node top(in0 : real) returns (x0, x1 : real);
let
    assert(in0 >= -1. and in0 <= 1.);
    x0 = 0. -> 0.499 * pre x0 - 0.05 * pre x1 + in0;
    x1 = 0. -> 0.01 * pre x0 + pre x1;
tel
```

- ▶ Kleene iteration using support function : **60637 iterations**
- ▶ Accelerated Kleene iteration using support function :  
**76 iterations**



Program			Polyhedra with widening		$\mathbb{P}_\Delta^\sharp$	
Name	V	\Delta	t(s)	Bounded	t(s)	Iteration
lead_leg_controller	5	350	TO	-	4.815 (1m33.356s)	76 (60637)
lp_iir_9600_2	6	372	0.12	No	0.023	47
lp_iir_9600_4	10	500	TO	-	0.186	100
lp_iir_9600_4_elliptic	10	500	TO	-	0.471	276
lp_iir_9600_6_elliptic	14	692	TO	-	3.636	702
bs_iir_9600_12000_10_chebyshev	22	1268	TO	-	53.986	2391
non_linear_ODE	3	316	TO	-	0.059	13

### non\_linear\_ODE program:

```

begin
dt = 0.01
gamma = 0.1
-2 <= x <= 2
while (true) do
    x1 = x1 + dt * ( -(1 + gamma * x2 * x2) * x1 )
    x2 = x2 + dt * (-0.5 * x2 * (1 - gamma * x1 * x1) + 2 * x3)
    x3 = x3 + dt * (-(1 - gamma * x1) * 2 * x2 - 0.5 * x_3 )
done;
end

```



## Conclusion

- ▶ We develop a new numerical abstract domain based on support function.
- ▶ Our abstract domain depends on a set of finite directions.
- ▶ In the case where the fixpoint computation terminates, it is obtained in a polynomial time.

## Perspectives

- ▶ According to the program to analyse, defines a relevant set of directions.
- ▶ Improve the meet operators, to avoid the loose of precision.